REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
21-12-2012	FINAL	SEPT 30, 2009 - SEPT 29, 2012

4. TITLE AND SUBTITLE
(DEPSCOR FY09) OBFUSCATION AND DEOBFUSCATION OF INTENT OF COMPUTER PROGRAMS

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA9550-09-1-0715

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
ARUN LAKHOTIA
VIR V PHOHA

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
UNIVERSITY OF LOUISIANA AT LAFAYETTE, LAFAYETTE, LA
LOUISIANA TECH, RUSTON, LA

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AF OFFICE OF SCIENTIFIC RESEARCH
875 N. RANDOLPH ST. ROOM 3112
ARLINGTON VA 22203
Dr. Robert Herklotz/RSL

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR/PKR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
afrl-osr-va-tr-2012-1798

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This research aimed at developing a theoretical framework to predict the next obfuscation (or deobfuscation) move of the adversary, with the intent of making cyber defense proactive. The goal was to understand the relationship between obfuscation and deobfuscation techniques employed in malware offense and defense. The strategy was to build upon previous work of Giacobazzi and Dalla Preda on modeling obfuscation and deobfuscation as abstract interpretations, further that effort by developing an analytical model of the best obfuscation with respect to a deobfuscator. In addition, this research aimed at developing cost models for obfuscation and deobfuscations. The key findings of this research include: a theoretical model of computing the best obfuscation for a deobfuscator, a method for context-sensitive analysis of obfuscated code, a method for learning obfuscation transformations used by a metamorphic engine, several insights into the use of machine learning in deobfuscation, and game-theoretic models of certain scenarios of offense-defense games in software protection.

15. SUBJECT TERMS
Software Obfuscation; Software Deobfuscation; Software Protection; Abstract Interpretation; Malware Analysis; Game Theory; Metamorphism; Polymorphism; Malware Attribution.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU	11	ARUN LAKHOTIA
					19b. TELEPHONE NUMBER *(Include area code)* (337) 482-6766

INSTRUCTIONS FOR COMPLETING SF 298

1. REPORT DATE. Full publication date, including day, month, if available. Must cite at least the year and be Year 2000 compliant, e.g. 30-06-1998; xx-06-1998; xx-xx-1998.

2. REPORT TYPE. State the type of report, such as final, technical, interim, memorandum, master's thesis, progress, quarterly, research, special, group study, etc.

3. DATES COVERED. Indicate the time during which the work was performed and the report was written, e.g., Jun 1997 - Jun 1998; 1-10 Jun 1996; May - Nov 1998; Nov 1998.

4. TITLE. Enter title and subtitle with volume number and part number, if applicable. On classified documents, enter the title classification in parentheses.

5a. CONTRACT NUMBER. Enter all contract numbers as they appear in the report, e.g. F33615-86-C-5169.

5b. GRANT NUMBER. Enter all grant numbers as they appear in the report, e.g. AFOSR-82-1234.

5c. PROGRAM ELEMENT NUMBER. Enter all program element numbers as they appear in the report, e.g. 61101A.

5d. PROJECT NUMBER. Enter all project numbers as they appear in the report, e.g. 1F665702D1257; ILIR.

5e. TASK NUMBER. Enter all task numbers as they appear in the report, e.g. 05; RF0330201; T4112.

5f. WORK UNIT NUMBER. Enter all work unit numbers as they appear in the report, e.g. 001; AFAPL30480105.

6. AUTHOR(S). Enter name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. The form of entry is the last name, first name, middle initial, and additional qualifiers separated by commas, e.g. Smith, Richard, J, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES). Self-explanatory.

8. PERFORMING ORGANIZATION REPORT NUMBER. Enter all unique alphanumeric report numbers assigned by the performing organization, e.g. BRL-1234; AFWL-TR-85-4017-Vol-21-PT-2.

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES). Enter the name and address of the organization(s) financially responsible for and monitoring the work.

10. SPONSOR/MONITOR'S ACRONYM(S). Enter, if available, e.g. BRL, ARDEC, NADC.

11. SPONSOR/MONITOR'S REPORT NUMBER(S). Enter report number as assigned by the sponsoring/ monitoring agency, if available, e.g. BRL-TR-829; -215.

12. DISTRIBUTION/AVAILABILITY STATEMENT. Use agency-mandated availability statements to indicate the public availability or distribution limitations of the report. If additional limitations/ restrictions or special markings are indicated, follow agency authorization procedures, e.g. RD/FRD, PROPIN, ITAR, etc. Include copyright information.

13. SUPPLEMENTARY NOTES. Enter information not included elsewhere such as: prepared in cooperation with; translation of; report supersedes; old edition number, etc.

14. ABSTRACT. A brief (approximately 200 words) factual summary of the most significant information.

15. SUBJECT TERMS. Key words or phrases identifying major concepts in the report.

16. SECURITY CLASSIFICATION. Enter security classification in accordance with security classification regulations, e.g. U, C, S, etc. If this form contains classified information, stamp classification level on the top and bottom of this page.

17. LIMITATION OF ABSTRACT. This block must be completed to assign a distribution limitation to the abstract. Enter UU (Unclassified Unlimited) or SAR (Same as Report). An entry in this block is necessary if the abstract is to be limited.

(DEPSCOR FY09) Obfuscation and Deobfuscation of Intent of Computer Programs

Arun Lakhotia
University of Louisiana at Lafayette, Lafayette, LA

Vir V. Phoha
Louisiana Tech University, Ruston, LA

FINAL REPORT FOR THE PERIOD SEPTEMBER 30, 2009 TO SEPTEMBER 29, 2012.
GRANT NUMBER: FA9550-09-1-0715

December 21, 2012

Abstract

This research aimed at developing a theoretical framework to predict the next obfuscation (or deobfuscation) move of the adversary, with the intent of making cyber defense proactive. More specifically, the goal was to understand the relationship between obfuscation and deobfuscation techniques employed in malware offense and defense. The strategy was to build upon previous work of Giacobazzi and Dalla Preda on modeling obfuscation and deobfuscation as abstract interpretations. It furthers that effort by developing an analytical model of the best obfuscation with respect to a deobfuscator. In addition, this research aimed at developing cost models for obfuscation and deobfuscations.

The key findings of this research include: a theoretical model of computing the best obfuscation for a deobfuscator, a method for context-sensitive analysis of obfuscated code, a method for learning obfuscation transformations used by a metamorphic engine, several insights into the use of machine learning in deobfuscation, and game-theoretic models of certain scenarios of offense-defense games in software protection.

Table of Contents

1 Research Team

The research effort was led by Arun Lakhotia and Andrew Walenstein of University of Louisiana at Lafayette, Vir Phoha of Louisiana Tech, and Bin Mai of Northwestern University. Other researchers who participated in the effort are: Wu Feng, Richard Greechie, and Enamul Karim of LaTech; Suresh Golconda and Anshuman Singh of UL Lafayette. The research also benefited from collaboration with Roberto Giacobazzi of University of Verona, Italy and Mila Dalla Preda of University of Bologna, Italy. Several graduate and undergraduate students participated in the research. Bin Mai moved to Bowie State, MD, in the first year of the project, and subsequently discontinued participation.

2 Research Outcomes

The research results can be summarized along three broad categories based on the underlying theories:

- Abstract Interpretation
- Machine Learning
- Game Theory

In the literature, code obfuscation and deobfuscation are treated as operations on individual programs. Our works using abstract interpretation are based on such models of obfuscation and deobfuscation. In addition, we also use machine learning (ML) to model obfuscation and deobfuscation as operations on collections. The ML-based approaches better represent the real-world, where intractable problems are addressed using probabilistic models. Finally, we use game-theory to model the strategic trade-offs in an offense-defense game played out in a few specific scenarios.

Our investigation in obfuscation and deobfuscation has led to identification of a new method of attack on software that does not appear to have been studied thus far. In this attack, a program is made to attack against itself by taking control of its execution and changing its control flow.

The rest of this report summarizes our research results.

2.a Abstract Interpretation

2.a.i Theoretical Model of Potency of Obfuscation (Dalla Preda et al., 2012)

Wu Feng, a Post Doctoral Researcher supported by this grant, worked with PI Lakhotia and collaborators and discovered several interesting properties of possibly obfuscated programs under abstraction. As a program transformation, an obfuscation preserves some properties of a program. The largest set of properties preserved by an obfuscation defines its potency. Using Giacobazzi and Dalla Preda's abstract interpretation based model of obfuscation we show that the largest correct approximation of an obfuscation can be obtained using its 'residuated' approximation, which in turn can be computed by iterating over its 'shadow'.

This result is significant in that it provides a method to calculate the least obfuscation function that can defeat a deobfuscation function. Whereas previous work of Giacobazzi and Dalla Preda had created separate partial order lattices of obfuscators and deobfuscators, our result establishes links between the two lattices. The relationship established does not yet define a partial order, as it does not establish the best deobfuscation function for a given obfuscation function.

2.a.ii Context Sensitive Analysis of Binaries (Boccardo et al. 2009; Lakhotia et al., 2010a and 2010b)

When it comes to analyzing programs in the context of this project, the analysis is performed on binary executables. Thus far, analyses of binaries have been modeled on analyses of source code. In the prevalent approach, a binary is disassembled, split into procedures, a control flow graph is constructed for each procedure, and then classic interprocedural analysis is applied. While this approach has enabled the use of prior experience in analyzing binaries, it has significant limitations. A majority of the limitations stem from the fact that binary (or assembly) programs do not have primitives for code or data encapsulation. As a result, the methods for finding procedure and code block boundaries are at best heuristic.

We have developed a method for context sensitive analysis of binaries without requiring procedural decomposition of the binary. The method is resilient to class call and return obfuscation attacks. Our method is based on the insight that classic interprocedural analysis depends on call and return instructions to achieve two tasks – creation of a new context and transfer of control. Programs can be obfuscated by using a separate set of instructions to achieve each. Our method, based on abstract interpretation, decouples the two tasks. As a result, the method does not depend on encapsulation of code into procedures to perform context sensitive analysis.

2.a.iii Learning Metamorphism (Unpublished)

Metamorphism is used to transform code so as to reduce or remove static footprints that can be used to relate two versions of the program. Metamorphism is used by malware to evade detection from static signature based detectors. Its use has increased as the malware distribution mechanism has moved to the web through infected sites. In this use a site is hacked so as to distribute malware to users connecting to the site. Instead of distributing the same copy of malware to each user, the server transforms the binary and distributes metamorphic variants.

In prior work, we showed that if the transformation rules of a metamorphic engine are known, then the variants it produces can be transformed back to a normal form with a reasonable confidence. We have now developed a method to extract transformation rules by syntactic and semantic comparison of malware variants generated by an engine. Our method combines abstract interpretation and data mining. It uses abstract (symbolic) interpretation to compute the semantics of individual blocks of programs. Code blocks from two variants with matching semantics are then paired. The actual code of the paired blocks is compared and a 'diff' is computed. The differences in the paired segments of the two variants provides a set of candidate transformed segments, which are then filtered again to keep only those pairs with equivalent semantics. The matching pair of transformed code is then generalized to create a set of general rules.

We have used our method to successfully extract the transformation rules used by the Win32.Evol metamorphic virus. A manuscript describing the results is in preparation.

2.b Machine Learning

2.b.i Detection of Anomalous Behavior in Computer Programs (Kanaskar et al., 2012)

Co-PI Phoha and his collaborators introduced techniques and formalisms of dynamical system theory into analysis of program behavior via system calls to detect code injections into an application's execution space. They consider a program as a blackbox dynamical system whose internals are not known, but whose output can be observed. The blackbox system observable in the proposed model is the system calls a program makes. The collected system calls are treated as signals which are used to reconstruct the system's phase space. Then, by using techniques from dynamical system theory, they quantify the amount of complexity of the system's (program's) behavior. The change in the behavior of a compromised system is detected as anomalous behavior compared to the baseline established from a clean program.

2.b.ii Mimicking Biometric Behavior by Computer Programs (Rehman et al., 2011; Serwadda et al., 2011)

We have investigated the strength of password and keystroke dynamics (KD) based co-authentication systems against synthetic attacks by bots and malware, and we showed that regardless of password length there is a category of users whose KD templates are critically vulnerable to such attacks. Co-PI Phoha and his students, Knandakar Rahman and Abdul Serwadda, have shown how malicious computer programs can mimic human behavior and defeat behavioral signature based authentication and verification systems using (i) individual user's snooped behavior, and (ii) global knowledge of users' behavior. These studies are conducted in the context of keystroke dynamics based biometrics systems.

2.b.iii Information Fusion to See Through Obfuscation (LeDoux et al., 2012)

In this research we studied how to take advantage of the limitations of obfuscations. It has previously been established that there can be no perfect obfuscation. In other words, one cannot obfuscate a program such that it can hide all of its interesting properties from all possible analyzers (deobfuscators). Hence, obfuscations are targeted against specific methods of deobfuscations. We studied how to use multiple, independent sensors to detect a program that is attempting to evade detection by dynamic analyzers. Our study indicates that fusing the sensor data using disjoint union amplifies the signal that indicates obfuscation.

The key significance of our result is that information fusion, if done correctly, can turn an obfuscation attempt against the adversary. The attempted obfuscation itself becomes a signal that in fact aids in detection of evasive malware.

2.b.iv Concept Drift in Malware Families (Singh et al., 2012a)

Machine learning methods are predicated on having training data representative of the population under study. One weakness this leads to is that when the population changes, the training data utilized must also change, and any concepts learned through machine learning must be relearned. This problem is compounded when a human and machine learner have an adversarial relationship as exists between

malware authors and machine learning methods used to detect malware. Thus, a fundamental problem in using machine learning for malware detection is deciding if and when to update training data and retrain machine learners.

This work focused on developing a sensor for measuring concept drift in malware families. The term concept drift is used to describe the situation where the statistical properties of the population under study changes. Previous works on measuring drift, done in the context of recommendation systems, did so by tracking individual features. These previous measures cannot be applied in domains where the feature sets are very large, as is the case for documents and programs. We have defined a measure for concept drift using the changes in similarity of malware variants with respect to a reference variant. The insight is that, over time, malware within a family will become different from this reference variant. The change in similarity can then be used to determine drift. The method was calibrated using successive variants of benign programs from open source repository, and then evaluated using malware data from the wild.

2.b.v Detecting Programs Generated by the Same Automated Engine (Chouchane et al, 2012)

Software engineering and software protection have conflicting goals. Whereas software engineering requires a program to be easily understood and modified, software protection requires the opposite. This implies that large complex systems cannot be developed from the ground up in manner that the intellectual property they contain is also protected. If the programs were difficult to comprehend, the programmers would not be able to debug, test, or verify them. Hence, software protection is by necessity performed by automated tools after a system has been developed.

An important step in breaking the protection is to determine the method of protection used, which in turn can be modeled as the problem of determining the automated engine used to generate a program. Our research introduced two forms of the "engine attribution" problem. The simple case of the problem is when the engine(s) whose creations are to be recognized are known. A more complex problem is when only programs created by one or more engines are known, but the engine itself is not accessible. The paper presents machine learning methods for computing approximate solutions for both the problems and presents results from experimental study.

2.c Game Theory

2.c.i Information Exchange for Packers (Singh et al., 2011)

Co-PI Lakhotia and graduate student Singh participated in the IEEE Malware Working Group, a consortium of various anti-malware companies, chartered with the task of creating a cooperative infrastructure to aid in detecting packed malware. The effort of this working group has led to the creation of the IEEE Taggant Malware System, a cooperative exchange between anti-malware vendors and packer (software protection) developers. Packers use obfuscation techniques to deter inspection of the system they are protecting. While this is desirable for protecting intellectual property, the mechanism also aids malware in evasion. This group faced the challenge of developing a system that

would help preventing malware from using packing for evasion, without violating protection of intellectual property.

Co-PI Lakhotia participated in the creation of the concept of software taggant. Drawing analogy from taggants used for tracking explosives, a software taggant aids in tracking a packed program to its creator, without revealing its internal code. The IEEE Taggant Malware System uses an infrastructure similar to code signing, but with a slight twist. A taggant is a signature introduced by a packer, and it contains information sufficient to identify the developer of the software packed.

Besides participating in the development of the IEEE Taggant Malware System, the two researchers also developed a game-theoretic model for assessing the incentive needed for software protection vendors to participate in this exchange in the presence of the 'moral hazard' that by such participation they stand to lose business from malware authors. The phrase 'moral hazard' captures the situation in the principal-agent problem where the agent, in our case the software protection vendor, is expected to work on the behalf of the principal, in our case the anti-malware vendors, and where the interests of the agent and the principal are not aligned. Our research highlighted the existence of the moral hazard and developed an economic model to compute the incentive the principal (anti-malware vendors) may provide to the agent (software protection vendors) such that it is not in the interest of the agent to pursue a business (selling non-taggant compliant product to malware authors) that is detrimental to the interest of the principal.

2.c.ii Game Theoretic Strategy Against Social Network Bots Inferring Users' Identities and Preferences (Chen et al., 2012)

Co-PI Dr. Phoha and his student Jundong Chen analyzed several game theoretical models, including their evolutionary extensions, to find the optimal strategy for the disclosure of user profile attributes in social networks from game-theoretic perspectives. Revealing attributes increases users' chances of finding friends and attracting new ones. However, excessive disclosure of attributes may enable inference of their identities and preferences by malicious programs or users. Results of the analysis indicate that users of social networks should adopt maximum privacy settings if such risk is involved, despite the intensity of their motivation to reveal attributes.

2.c.iii Deployable Performance for Classifiers in Adversarial Environment (Singh et al., 2012b).

Many security systems are supervised classification systems. They are classically evaluated by generalizing their performance for unseen data from their performance on known samples from the population that the system is to classify. Generalization performance is estimated from test performance using confidence intervals (CIs), which depend on the method used for determining test performance: holdout, cross-validation, or bootstrap. For instance, the generalization performance of a classification model may be measured using the ratio of false positives on a k-fold cross validation test.

We argue that generalization performance may not be a good determinant of the performance of a classifier deployed in an adversarial environment because the population characteristic may change in response to deployment of the classifier. We present a model for deployable performance that 'prescribes' a generalization performance target for a security technology such that the expected cost to

the deployer is below a certain threshold. The deployable performance model factors in not just the cost of response to a security breach, but also the cost to the adversary for gaming the system by obfuscating the 'features' observed by the classification model.

2.d New Method of Deobfuscation

2.d.i Using a Program Against Itself (Miles et al., 2012)

There are two classic models of deobfuscation (or attack on software) – static analysis and dynamic analysis. In static analysis, the software is analyzed without executing it. In dynamic analysis, the software is executed, and analyzed by monitoring its internal state and its interaction with the external environment.

There is a specific situation that arises when analyzing botnet malware that challenges these two methods. A bot typically communicates with its master over a cryptographically secure channel. In addition, the bot may also contain code that is encrypted. Thus, an important step in analyzing or taking control of a bot is bypassing its encryption. This can be challenging for both static and dynamic analysis. The bots are often designed to connect to the master before activating their encryption or decryption logic. However, connecting to the master can reveal the presence of the analyst, and may give the bot master an ability to activate a self-destruct command.

One common method for breaking the encryption/decryption is to identify the encryption/decryption procedures, reconstruct them in a different program, apply them to decrypt the original binary, and then apply static and dynamic analysis on the decrypted program. This method of attack can be quite time consuming and is prone to error.

In the course of this research, we have discovered a method by which the code within a binary can be directly executed 'in situ.' This allows for using the decryption functions within a binary directly, but outside of the malware author's control. Thus, dynamic and static analysis on the binary of the relevant procedures can be performed without requiring that the cryptographic code be reconstructed in a separate program.

3 Publications

(Boccardo et al., 2009) Davidson R. Boccardo, Arun Lakhotia, A. Manacero Jr, and Michael Venable. Adapting call-string approach for x86 obfuscated binaries. *Simpósio Brasileiro em Segurança da Informaçao e de Sistemas Computacionais* (2009).

(Chen et al., 2012) Jundong Chen, Matthias R. Brust, Vir V. Phoha and Ankunda R. Kiremire. A Model for Analyzing Privacy Settings of Social Networks from a Game theoretical Perspective. *Special Issue of Springer CASM on Agent-based Modeling* (submitted), 2012.

(Chouchane, 2012) Mohamed Chouchane, Natalia Stakhanova, Andrew Walenstein, and Arun Lakhotia. Detecting Machine-Morphed Malware Variants Via Authorship Attribution, *Journal in Computer Virology*, 2012 (to appear).

(Kanaskar et al., 2012) Nitin Kanaskar, Ramzi Seker, Jiang Bian, Vir Phoha. Dynamical System Theory for Detection of Anomalous Behavior in Computer Programs, I*EEE Transactions on SMC, Part C, 2012 (accepted)*

(Lakhotia et al., 2010) Arun Lakhotia, Davidson Boccardo, Anshuman Singh, Aleardo Manacero, Jr. Context-Sensitive Analysis without Calling-Context. *Journal of Higher-Order and Symbolic Computation*, Springer. 23 (3) 275-313, 2010. DOI: 10.1007/s10990-011-9080-1.

(Lakhotia et al., 2010) Arun Lakhotia, Davidson R. Boccardo, Anshuman Singh, and Aleardo Manacero, Jr. Context-sensitive analysis of obfuscated x86 executables. In Proceedings of the 2010 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (Madrid, Spain, January 18 - 19, 2010). PEPM '10. ACM, New York, NY, 131-140. DOI= http://doi.acm.org/10.1145/1706356.1706381

(Ledoux et al., 2012) Charles Ledoux, Andrew Walenstein and Arun Lakhotia. Improved Malware Classification Through the Use of Differences in Sensor Outputs, *Information Systems Technology and Management: Communications in Computer and Information Science*, Springer Berlin, Volume 285, Part 7, 360-371, 2012. DOI: 10.1007/978-3-642-29166-1_32.

(Mai et al., 2010) Bin Mai, Nirup M. Menon, Sumit Sarkar, No Free Lunch: Price Premium for Privacy Seal-Bearing Vendors, *Journal of Management Information Systems*, 27 (2), Fall 2010, 189-212, DOI:10.2753/MIS0742-1222270206.

(Miles et al., 2012) Craig Miles, Arun Lakhotia, and Andrew Walenstein. In Situ Reuse of Logically Extracted Functional Components. *Journal in Computer Virology*. Springer Paris, 8 (3): 73-84, 2012. DOI: 10.1007/s11416-012-0167-y.

(Dalla Preda et al., 2012) Mila Dalla Preda, Wu Feng, Roberto Giacobazzi, Richard Greechie and Arun Lakhotia. Twisting Additivity in Program Obfuscation, *Information Systems Technology and Management: Communications in Computer and Information Science*, Springer Berlin, Volume 285, Part 7, 336-347, 2012. DOI: 10.1007/978-3-642-29166-1_30.

(Rahman et al., 2011) K. A. Rahman, K.S. Balagani, V. V. Phoha . Making impostor pass rates meaningless: A case of snoop-forge-replay attack on continuous cyber-behavioral verification with keystrokes, *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on* , vol., no., pp.31-38, 20-25 June 2011. doi: 10.1109/CVPRW.2011.5981729

(Serwadda et al., 2011) Abdul Serwadda, Vir V. Phoha, and Ankunda Kiremire. 2011. Using global knowledge of users' typing traits to attack keystroke biometrics templates. In *Proceedings of the thirteenth ACM multimedia workshop on Multimedia and security* (MM&Sec '11). ACM, New York, NY, USA, 51-60. DOI=10.1145/2037252.2037263 http://doi.acm.org/10.1145/2037252.2037263

(Singh et al., 2010) Anshuman Singh, Arun Lakhotia, Andrew Walenstein. Malware Anti-Malware Games. 5th International Conference on Information Warfare 2010, April 8-19, 2010. Dayton, OH, pp. 319-327.

(Singh and Lakhotia, 2011) Anshuman Singh and Arun Lakhotia. Game-theoretic Design of an Information Exchange Model for Detecting Packed Malware. *Malicious and Unwanted Software (MALWARE), 2011 6th International Conference on* , vol., no., pp.1-7, 18-19 Oct. 2011. doi: 10.1109/MALWARE.2011.6112319

(Singh et al., 2012a) Anshuman Singh, Andrew Walenstein, and Arun Lakhotia. 2012. Tracking concept drift in malware families. In *Proceedings of the 5th ACM workshop on Security and artificial intelligence* (AISec '12). ACM, New York, NY, USA, 81-92. DOI=10.1145/2381896.2381910.

(Singh et al., 2012b) Anshuman Singh, Sumi Singh, Andrew Walenstein and Arun Lakhotia. Deployable classifiers for malware detection, *Information Systems Technology and Management: Communications in Computer and Information Science*, Springer Berlin, Volume 285, Part 7, 384-395, 2012. DOI: 10.1007/978-3-642-29166-1_34.

4 Students Participating in the Research

1. Jun-Dong Chen (MS, Mathematics, Louisiana Tech, Nov 2012)
   (PhD., Computational Analysis and Modeling, Louisiana Tech, In Progress)

2. Daniel J. Hefner (B.S., Computer Science, UL Lafayette, May 2012)

3. Shafaeat Hossain (MS, Computer Science, Louisiana Tech, Nov 2012)
   (PhD., Computational Analysis and Modeling, Louisiana Tech, In Progress)

4. David Irakiza (MS, Mathematics, Louisiana Tech, Aug 2012)
   (PhD., Computational Analysis and Modeling, Louisiana Tech, In Progress)

5. Ankunda R. Kiremire (MS, Mathematics, Louisiana Tech, Aug 2012)
   (PhD., Computational Analysis and Modeling, Louisiana Tech, In Progress)

6. Charles LeDoux (M.S., Computer Science, UL Lafayette, May 2011)
   (Ph.D., Computer Science, University of Louisiana at Lafayette, In Progress)

7. Craig Miles (Ph.D. Computer Science, UL Lafayette, In Progress)

8. Chris Parich (B.S., Computer Science, UL Lafayette, In Progress)

9. Abena Primo (MS, Computer Science, Louisiana Tech, May 2012)
   (PhD., Computational Analysis and Modeling, Louisiana Tech, In Progress)

10. K. Abir Rahman (PhD, Computer Science, LaTech, in progress)

11. Sumi Singh (Ph.D., Computer Science, UL Lafayette, in progress)

12. Abdul Serwadda (MS, Computer Science, Louisiana Tech, Nov 20 2012)
(PhD., Computational Analysis and Modeling, Louisiana Tech, In Progress)

13. Anshuman Singh (Ph.D., Computer Science, UL Lafayette, June 2012)

14. Matthew Wallace (B.S., Computer Science, UL Lafayette, In Progress)

15. Zibo Wang (MS, Computer Science, Louisiana Tech, Aug 2012)
(PhD., Computational Analysis and Modeling, Louisiana Tech, In Progress)

5 Patents and Inventions

None